Eclipse Plug-in Development Research Project

Report 9

UNIVERSITY AT BUFFALO

August 20, 2015

Authored by: Chern Yee Chua

Eclipse Plug-in Development Research Project

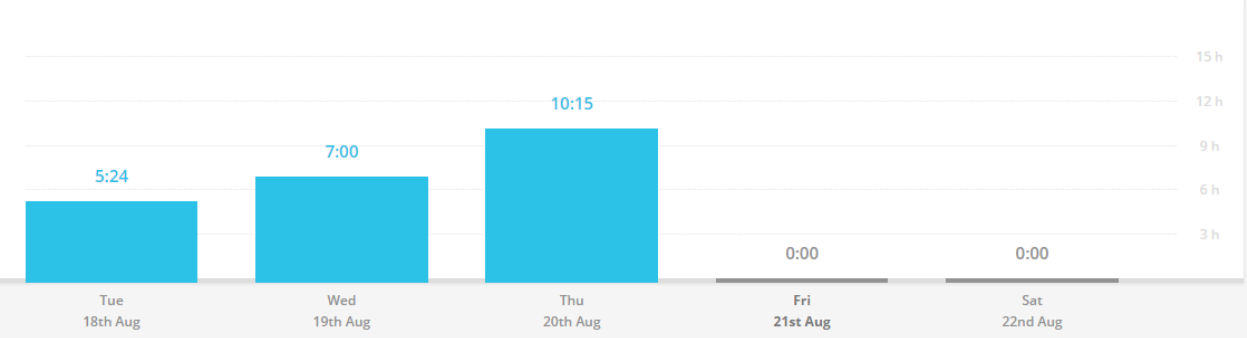
Report 9

Introduction

I am very excited to show what I have done for the project this week. After many hours of testing and debugging, I finally got everything worked out the way I have expected. More will be discussed in the section below.

Synopsis

I spend more than 20 hours on this project for this week.



Eclipse Plugin	22:39:52
Cleaning up code and creating doc	4:47:52
Finally fix the comments	2:40:00
Fixing list item issue when adding or removing items	5:24:00
Fixing some bugs and issues	2:48:00
Successfully fixing those messed up items	7:00:00

Explanation of content

Finally, I have fixed all the known issues from earlier. This plugin program now performs exactly the way we want. Issues such as adding or removing lines that causes misunderstanding of element changes, the duplicated elements, the comment lines' issue, changing element in the same line, and etc. have all been taken care of. The rest element changed events such as adding new file or package is really a piece of cake compared to those above.

The one took me the longest to fix is the duplicate elements since the only differences is the line number and that might be just come from a shifted block. I have tried many ways, such as finding the duplicate elements and checking the line number difference whether it is consistent to consider it as a shifted block of code but that takes too much work and it is way too complicated than it should be. My final solution is to mark the duplicated element as "duplicate X" where "X" is the number of occurrence of the duplicate element. By that way, each element is unique and hence it makes the comparisons way easier.

The other issue is with understanding the element line. The information from the console is not enough even with the given line number of the code. Hence, I add the node's parent name into the element line so that we will get a better sense of what element is.

In order to understand which file the student is working on, the class name together with time stamp will be produced in certain amount of time. And whenever the student switches or works on other file, the system will give notice that the student has switched to other file.

Adding or removing java file, package, java project or anything under JavaModel will be notified in console as well.

I clean up the code and add some brief explanation of what each function does. I also upload my work on Github. Check out the link below for the source code:
https://github.com/issacchern/ResearchPluginListener

Please feel free to test out and let me know what else I can improve!

Demonstration

This is just a simple demonstration of how this plugin works. There will be more testing and changing for a better outcome.

When I create a Java Project named IssacProject, and then create a package named issacPackage and then create a class called IssacClass.

```
-------Listener Initialization--------
2015.08.20 at 22:24:12 EDT
[ADDED JAVA_PROJECT]: IssacProject (not open)
[ADDED PACKAGE_FRAGMENT]: issacPackage (not open) [in src [in IssacProject]]
[ADDED COMPILATION_UNIT]: IssacClass.java (not open) [in issacPackage [in src [in IssacProject]]]
----- NOTICE: USER HAS SWITCHED TO OTHER WORKING FILE! -----
package issacPackage;
public class IssacClass {
}

[issacPackage.IssacClass] 2015.08.20 at 22:24:59 EDT
```

Adding variables int a, b, c, d, e and also a method named method with local variable [int a]. Noticing that the [int a] at line 5 and 13 has different parent's node name, that might be a good indication of differentiating element.

```
[issacPackage.IssacClass] 2015.08.20 at 22:27:37 EDT
[ADDED LINE] : 5 $ [FIELD_DECLARATION] (int) | (a) |
[ADDED LINE] : 6 $ [FIELD_DECLARATION] (int) | (b) |
...
[ADDED LINE] : 7 $ [FIELD_DECLARATION] (int) | (c) |
[ADDED LINE] : 8 $ [FIELD_DECLARATION] (int) | (d) |
[ADDED LINE] : 9 $ [FIELD_DECLARATION] (int) | (e) |
...
[ADDED LINE] : 11 $ [METHOD_DECLARATION] keyword (public) | constructor (false) | (void) | (method) |
...
[ADDED LINE] : 13 $ [VARIABLE_DECLARATION_STATEMENT] (int) | (a) |
...
```

Changing [int a] at line 13 into [int a = 2]. As you can see, there is | token (2) | at the end of the new line and that is the indication of elements being updated with displaying the remove line of old element together. Noticed that there is a line with class name and the time stamp, it is triggered in certain amount of time. Feel free to change or remove it at ease.

```
[issacPackage.IssacClass] 2015.08.20 at 22:31:29 EDT
[ADDED LINE] : 13 $ [VARIABLE_DECLARATION_STATEMENT] (int) | (a) | token (2) |
[REMOVED LINE] : 13 $ [VARIABLE_DECLARATION_STATEMENT] (int) | (a) |
...
```

Let's try something more complicated.

```java
int a = 2;

if( a == 2){
    for(int x = 0 ; x < 2 ; x++){
        x += 2;
    }
}
```

```
...
[ADDED LINE] : 13 $ [VARIABLE_DECLARATION_STATEMENT] (int) | (a) | token (2) |
[ADDED LINE] : 15 $ [INFIX_EXPRESSION] (a) | operator (==) | token (2) |
[ADDED LINE] : 16 $ [VARIABLE_DECLARATION_EXPRESSION] (int) | (x) | token (0) | (x) | operator (<) | token (2) | (x) | operator (++) |
...
[issacPackage.IssacClass] 2015.08.20 at 22:38:08 EDT
[ADDED LINE] : 17 $ [ASSIGNMENT] (x) | operator (+=) | token (2) |
...
```

As you can see, there is no issue with outputting to console. The issue is that whether I can interpret the code correctly. Or maybe I need to work on the parent's node name, probably returning grandparent's node name if the current node's name doesn't do any good.

Now let's add another class called CarlClass. As you can see, whenever the user switches to another file, the system will give notice ahead and display the structure of the code for the first change.

```
[ADDED COMPILATION_UNIT]: CarlClass.java (not open) [in issacPackage [in src [in IssacProject]]]
----- NOTICE: USER HAS SWITCHED TO OTHER WORKING FILE! -----
package issacPackage;
public class CarlClass {
}

[issacPackage.CarlClass] 2015.08.20 at 22:47:47 EDT
```

Of course, detecting and displaying the comment is not a big deal too.

```java
4  public class CarlClass {
5
6      // this is line comment
7
8      /*
9       * this is block comment
10      *
11      */
12
13     /**
14      * this is javadoc comment
15      *
16      *
17      */
18
19
20 }
```

```
...
[ADDED LINE] : 6 # [LINE_COMMENT] // this is line comment
[ADDED LINE] : 8 # [BLOCK_COMMENT]
/*
         * this is block comment
         *
         */
[ADDED LINE] : 13 # [JAVADOC]
/**
         * this is javadoc comment
         *
         *
         */
...
```

Removing block comment works the same way as removing other element.

```
[issacPackage.CarlClass] 2015.08.20 at 22:59:39 EDT
[REMOVED LINE] : 8 # [BLOCK_COMMENT]
/*
         * this is block comment
         *
         */
...
```

Conclusion

I am very happy and proud for what I have done for this plugin project in this week. All my hard work does pay off! I hope professor will like it as well!

Actually what got me most excited is to deploy the plugin on school's computers and see how students code. I have the feeling that there will be a lot of information in the output file because students most likely will get a lot of errors and things like that. I would be expected to see a lot of ADDED and REMOVED of the same incomplete lines.

Thank you for your time!